BÀI 7: SỰ PHÂN TÍCH NHÓM (TT)

I. Mục tiêu:

Sau khi thực hành xong, sinh viên nắm được phương pháp gom cụm phân cấp (hierarchical) thông qua 2 phương pháp sau:

- Phương pháp hợp nhất Bottom-up
- Phương pháp phân tách Top-down

II. Tóm tắt lý thuyết:

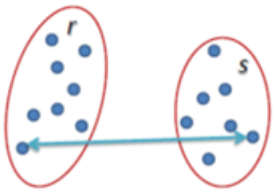
1. Phương pháp hợp nhất Bottom-up

Trong phương pháp này, các điểm dữ liệu được hợp nhất (agglomerative) liên tục thành các cụm ở mức cao hơn. Sự lựa chọn hàm mục tiêu thường quyết định việc trộn lại của các cụm. Thuật toán được phát biểu như sau:

Algorithm *AgglomerativeMerge*(Data: $\mathcal{D}$)
begin
  Initialize $n \times n$ distance matrix M using $\mathcal{D}$;
  repeat
    Pick closest pair of clusters i and j using M;
    Merge clusters i and j;
    Delete rows/columns i and j from M and create
      a new row and column for newly merged cluster;
    Update the entries of new row and column of M;
  until termination criterion;
  return current merged cluster set;
end

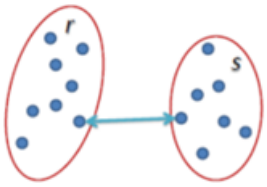
Hàm tiêu chuẩn liên kết (linkage criteria) được sử dụng để xác định khoảng cách giữa 2 cụm hoặc tập hợp các điểm. Các hàm liên kết phổ biến nhất được mô tả như sau:

- Liên kết tối đa hoặc đầy đủ (complete-linkage): Khoảng cách giữa hai cụm được định nghĩa là khoảng cách lớn nhất giữa 2 điểm trong mỗi cụm



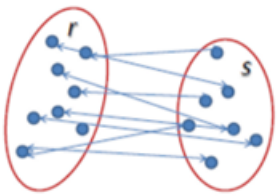
$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

- Liên kết đơn (single-linkage) hoặc tối thiểu: là khoảng cách nhỏ nhất giữa 2 điểm trong mỗi cụm.



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

- Liên kết trung bình: là khoảng cách trung bình giữa các điểm trong mỗi cụm.



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

- Liên kết tâm: là khoảng cách giữa tâm của mỗi cụm.
- Phương pháp phương sai tối thiểu của Ward: Nó giảm thiểu tổng phương sai trong cụm. Ở mỗi bước, cặp cụm có khoảng cách giữa các cụm tối thiểu được hợp nhất.

Ví dụ: Giả sử một giáo viên muốn chia 5 học sinh của lớp mình thành các nhóm khác nhau (không xác định rõ là có bao nhiêu nhóm), áp dụng gom cụm phân bậc ở đây và phân chia sinh viên thành các nhóm khác nhau.

ID	Điểm
1	10
2	7
3	28
4	20
5	35

Tạo ma trận khoảng cách M có kích thước 5×5 (sử dụng khoảng cách Euclidean):

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Các bước để thực hiện gom cụm phân bậc:

Bước 1: Tất cả các điểm là một cụm riêng lẻ

Màu sắc khác nhau ở đây đại diện cho các cụm khác nhau.

Bước 2: Tiếp theo, ta xét khoảng cách nhỏ nhất trong ma trận khoảng cách và hợp nhất các điểm có khoảng cách nhỏ nhất.

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Ở đây, khoảng cách nhỏ nhất là 3 và do đó chúng ta sẽ hợp nhất điểm 1 và 2:

Cập nhật lại ma trận khoảng cách:

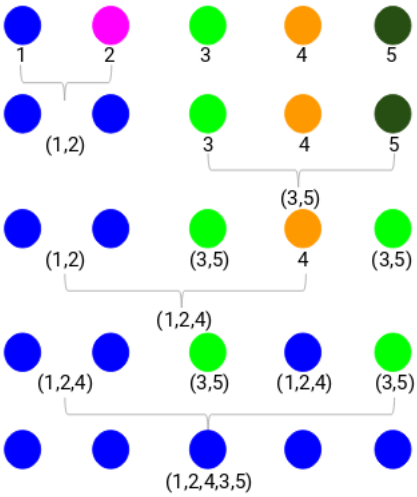
ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

Ở đây, hai cặp điểm (7, 10) ta lấy giá trị nhỏ nhất và tính toán lại ma trận khoảng cách cho các cụm này:

ID	Điểm
(1,2)	7
3	28
4	20
5	35

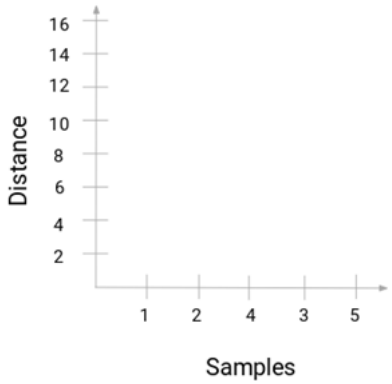
Bước 3: lặp lại bước 2 cho đến khi chỉ còn một cụm duy nhất.

Ta sẽ xét khoảng cách nhỏ nhất trong ma trận khoảng cách và sau đó hợp nhất cặp cụm gần nhất. Ta sẽ nhận được các cụm được hợp nhất như hình dưới đây sau khi lặp lại các bước sau:



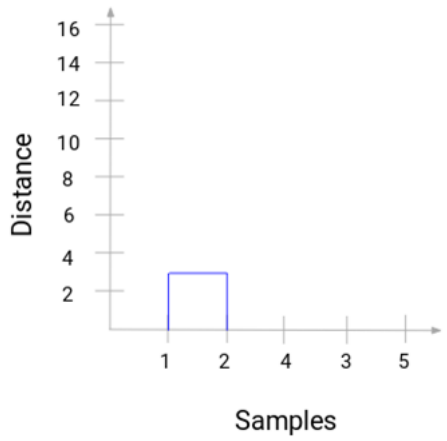
Để có được số lượng cụm cho gom cụm phân bậc, ta sử dụng một khái niệm có tên là Dendrogram.

Dendrogram là một sơ đồ dạng cây ghi lại các chuỗi hợp nhất hoặc chia tách. Bất cứ khi nào chúng ta hợp nhất hai cụm, một dendrogram sẽ ghi lại khoảng cách giữa các cụm này và biểu thị nó dưới dạng biểu đồ.

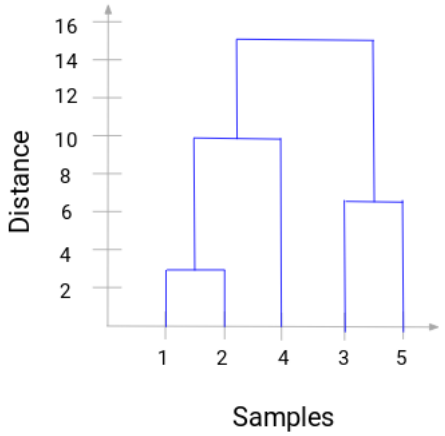


Các mẫu của tập dữ liệu trên trục x và khoảng cách trên trục y. Bất cứ khi nào hai cụm được hợp nhất, ta sẽ nối chúng trong chương trình dendro này và chiều cao của phép nối sẽ là khoảng cách giữa các điểm này.

Hợp nhất mẫu 1 và mẫu 2 và khoảng cách giữa hai mẫu này là 3. Đường thẳng đứng biểu thị khoảng cách giữa các mẫu này.

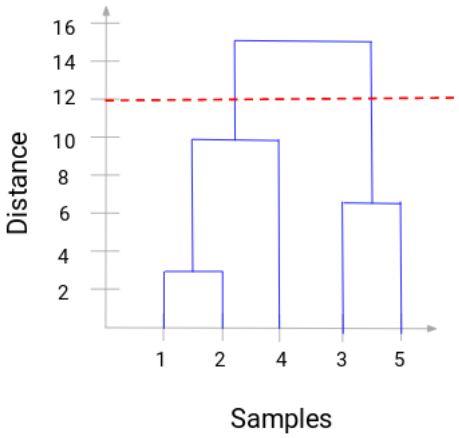


Tương tự, ta vẽ sơ đồ tất cả các bước đã hợp nhất các cụm và cuối cùng nhận được một chương trình dendro như sau:



Khoảng cách của các đường thẳng đứng trong dendrogram càng nhiều thì khoảng cách giữa các cụm đó càng nhiều.

Bây giờ, ta có thể đặt khoảng cách ngưỡng và vẽ một đường ngang (Nói chung, ta cố gắng đặt ngưỡng sao cho nó cắt đường thẳng đứng cao nhất). Hãy đặt ngưỡng này là 12 và vẽ một đường ngang:

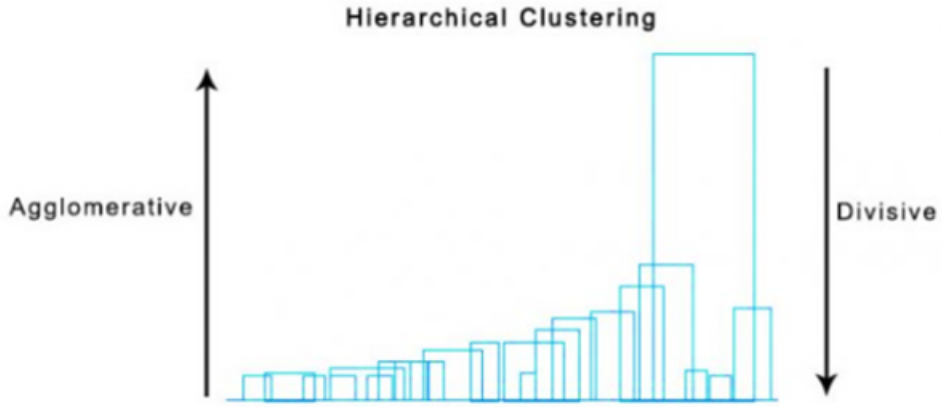


Số cụm sẽ là số đường thẳng đứng được cắt bởi đường được vẽ bằng ngưỡng. Trong ví dụ trên, do đường màu đỏ cắt 2 đường thẳng đứng nên ta sẽ có 2 cụm. Một cụm sẽ có một mẫu (1, 2, 4) và cụm kia sẽ có một mẫu (3, 5).

2. Phương pháp Top-down divisive

Phương pháp phân tách các cụm làm việc ngược lại với phương pháp hợp nhất Bottom-up. Thay vì bắt đầu với n cụm, ta bắt đầu với 1 cụm duy nhất và gán tất cả các điểm cho cụm này. Tại mỗi lần lặp, ta phân chia điểm xa nhất trong cụm và lặp lại quy trình này cho đến khi mỗi cụm chỉ chứa một điểm duy nhất. Thuật toán top-down được phát biểu như sau:

Algorithm *GenericTopDownClustering*(Data: $\mathcal{D}$, Flat Algorithm: $\mathcal{A}$)
begin
  Initialize tree $\mathcal{T}$ to root containing $\mathcal{D}$;
  repeat
    Select a leaf node L in $\mathcal{T}$ based on pre-defined criterion;
    Use algorithm $\mathcal{A}$ to split L into $L_1 \ldots L_k$;
    Add $L_1 \ldots L_k$ as children of L in $\mathcal{T}$;
  until termination criterion;
end



III. Nội dung thực hành:

1. Cài đặt thuật toán gom cụm phân cấp

- Đọc dữ liệu từ file "data.csv"

  (`https://www.dropbox.com/s/ikgzzr30ln6akk2/data.csv?dl=0`)

```
>>> import pandas as pd
>>> import seaborn as sns
>>> import scipy.cluster.hierarchy as shc
>>> import matplotlib.pyplot as plt
>>> from sklearn.decomposition import PCA
>>> from sklearn.cluster import AgglomerativeClustering
>>> path_to_file = 'D:\\Huynh\\DataMining_Lab\\data\\tuan7\\data.csv'
>>> customer_data = pd.read_csv(path_to_file)
>>> customer_data.shape
(200, 5)
>>> customer_data.columns
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
>>> customer_data.describe().transpose()
                        count    mean         std  ...    50%     75%    max
CustomerID              200.0  100.50   57.879185  ...  100.5  150.25  200.0
Age                     200.0   38.85   13.969007  ...   36.0   49.00   70.0
Annual Income (k$)      200.0   60.56   26.264721  ...   61.5   78.00  137.0
Spending Score (1-100)  200.0   50.20   25.823522  ...   50.0   73.00   99.0

[4 rows x 8 columns]
>>> customer_data.head()
   CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

- Chia cột "Age" thành 10 nhóm khác nhau $(15-20, 20-30, 30-40, 40-50, 50-60, 60-70)$

```
>>> intervals = [15, 20, 30, 40, 50, 60, 70]
>>> col = customer_data['Age']
>>> customer_data['Age Groups'] = pd.cut(x=col, bins=intervals)
>>> customer_data['Age Groups']
0      (15, 20]
1      (20, 30]
2      (15, 20]
3      (20, 30]
4      (30, 40]
         ...
195    (30, 40]
196    (40, 50]
197    (30, 40]
198    (30, 40]
199    (20, 30]
Name: Age Groups, Length: 200, dtype: category
Categories (6, interval[int64]): [(15, 20] < (20, 30] < (30, 40] < (40, 50] < (50, 60] < (60, 70]]
>>> customer_data.groupby('Age Groups')['Age Groups'].count()
Age Groups
(15, 20]    17
(20, 30]    45
(30, 40]    60
(40, 50]    38
(50, 60]    23
(60, 70]    17
Name: Age Groups, dtype: int64
```

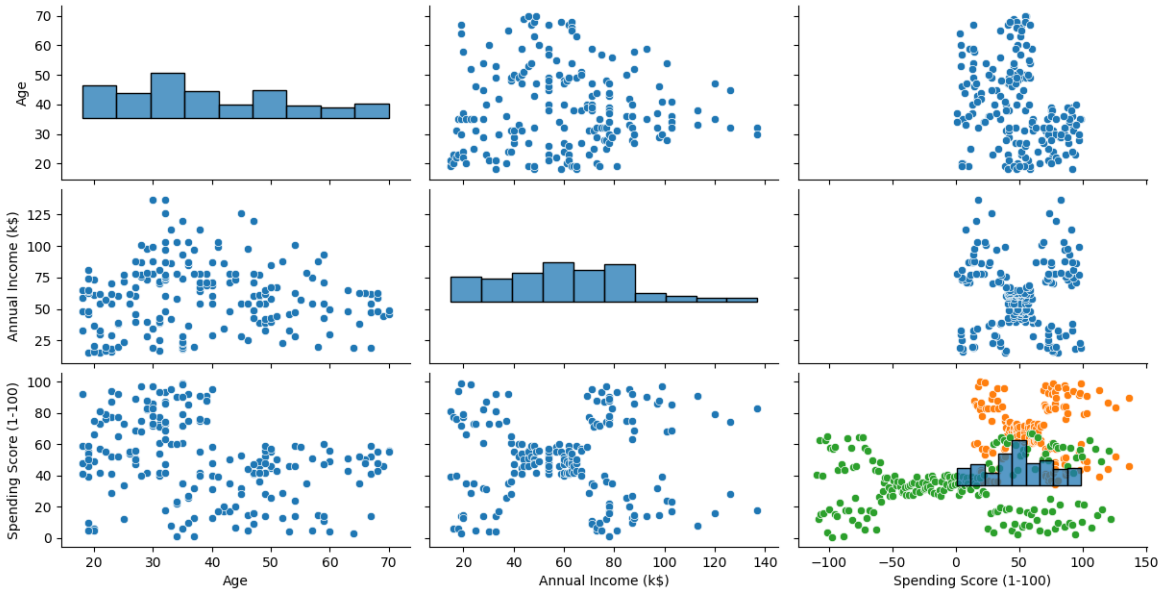
- Chuyển 2 cột Age và Genre thành dạng số

```
>>> customer_data_oh = pd.get_dummies(customer_data)
>>> customer_data_oh
     CustomerID  Age  ...  Age Groups_(50, 60]  Age Groups_(60, 70]
0             1   19  ...                    0                    0
1             2   21  ...                    0                    0
2             3   20  ...                    0                    0
3             4   23  ...                    0                    0
4             5   31  ...                    0                    0
..          ...  ...  ...                  ...                  ...
195         196   35  ...                    0                    0
196         197   45  ...                    0                    0
197         198   32  ...                    0                    0
198         199   32  ...                    0                    0
199         200   30  ...                    0                    0

[200 rows x 12 columns]
```

- Bỏ cột "CustomerID"

```
>>> customer_data = customer_data.drop('CustomerID', axis=1)
>>> sns.pairplot(customer_data)
<seaborn.axisgrid.PairGrid object at 0x00000227C6845AC8>
>>> sns.scatterplot(x=customer_data['Annual Income (k$)'],
                    y=customer_data['Spending Score (1-100)'])
<matplotlib.axes._subplots.AxesSubplot object at 0x00000227CA637848>
```

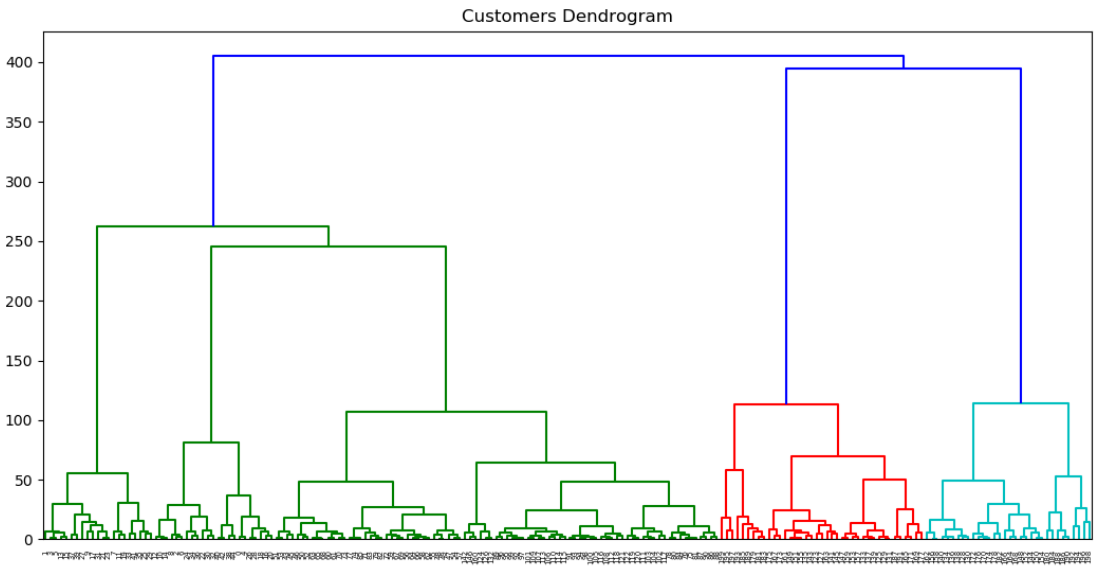


- Bỏ cột "Age" và vẽ dendrogram

```
>>> customer_data_oh = customer_data_oh.drop(['Age'], axis=1)
>>> customer_data_oh.shape
(200, 11)
>>> plt.figure(figsize=(10, 7))
<Figure size 1000x700 with 0 Axes>
>>> plt.title("Customers Dendrogram")
Text(0.5, 1.0, 'Customers Dendrogram')
>>> selected_data = customer_data_oh.iloc[:, 1:3]
>>> clusters = shc.linkage(selected_data,
            method='ward',
            metric="euclidean")
>>> shc.dendrogram(Z=clusters)
```

Squeezed text (263 lines).

```
>>> plt.show()
```

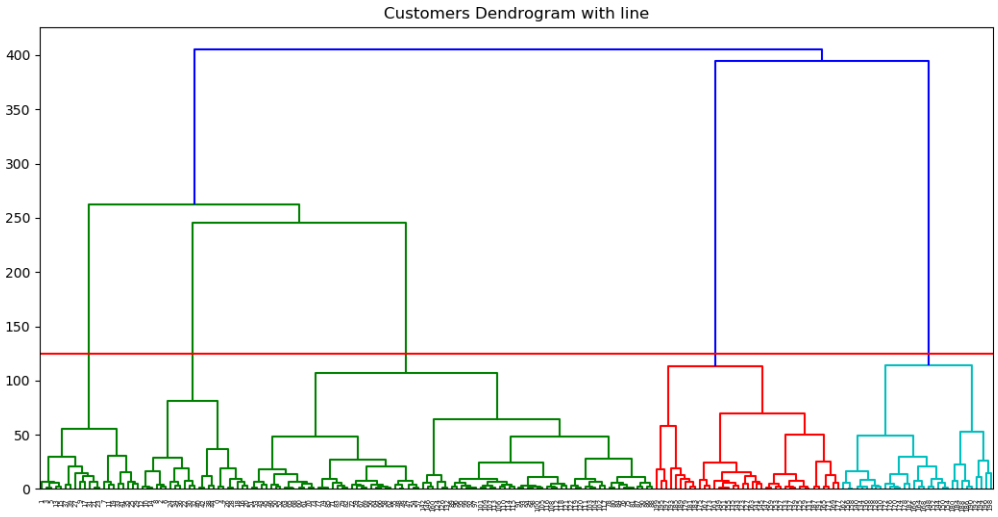


- Vẽ đường nằm ngang đi qua khoảng cách dài nhất

```
>>> plt.figure(figsize=(10, 7))
<Figure size 1000x700 with 0 Axes>
>>> plt.title("Customers Dendrogram with line")
Text(0.5, 1.0, 'Customers Dendrogram with line')
>>> clusters = shc.linkage(selected_data,
            method='ward',
            metric="euclidean")
>>> shc.dendrogram(clusters)
```

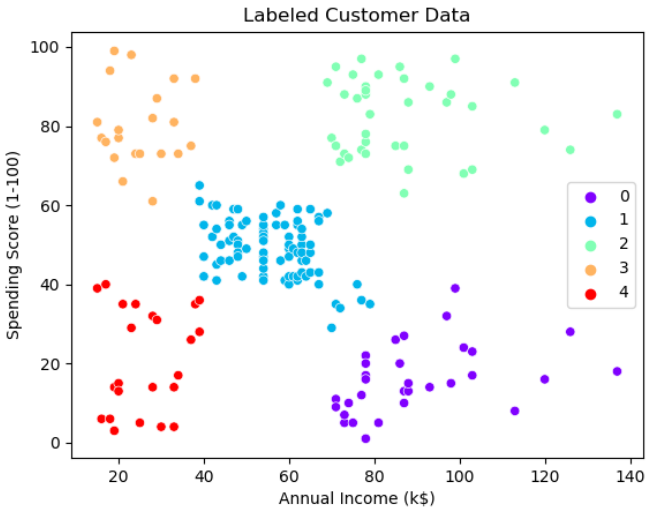
Squeezed text (263 lines).

```
>>> plt.axhline(y = 125, color = 'r', linestyle = '-')
<matplotlib.lines.Line2D object at 0x00000250F020E488>
>>> plt.show()
```



- Thực thi phân cụm với dữ liệu ban đầu

```
>>> clustering_model = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
>>> clustering_model.fit(selected_data)
AgglomerativeClustering(n_clusters=5)
>>> clustering_model.labels_
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int64)
>>> data_labels = clustering_model.labels_
>>> sns.scatterplot(x='Annual Income (k$)',
                    y='Spending Score (1-100)',
                    data=selected_data,
                    hue=data_labels,
                    palette="rainbow").set_title('Labeled Customer Data')
Text(0.5, 1.0, 'Labeled Customer Data')
>>> plt.show()
```



2. Yêu cầu

- Viết chương trình để thực thi thuật toán hierarchical (sử dụng phương pháp hợp nhất Bottom-up và phương pháp phân tách Top-down) với single-linkage.

- Trình bày tóm tắt phần code do em viết và so sánh với hàm có sẵn trong thư viện.